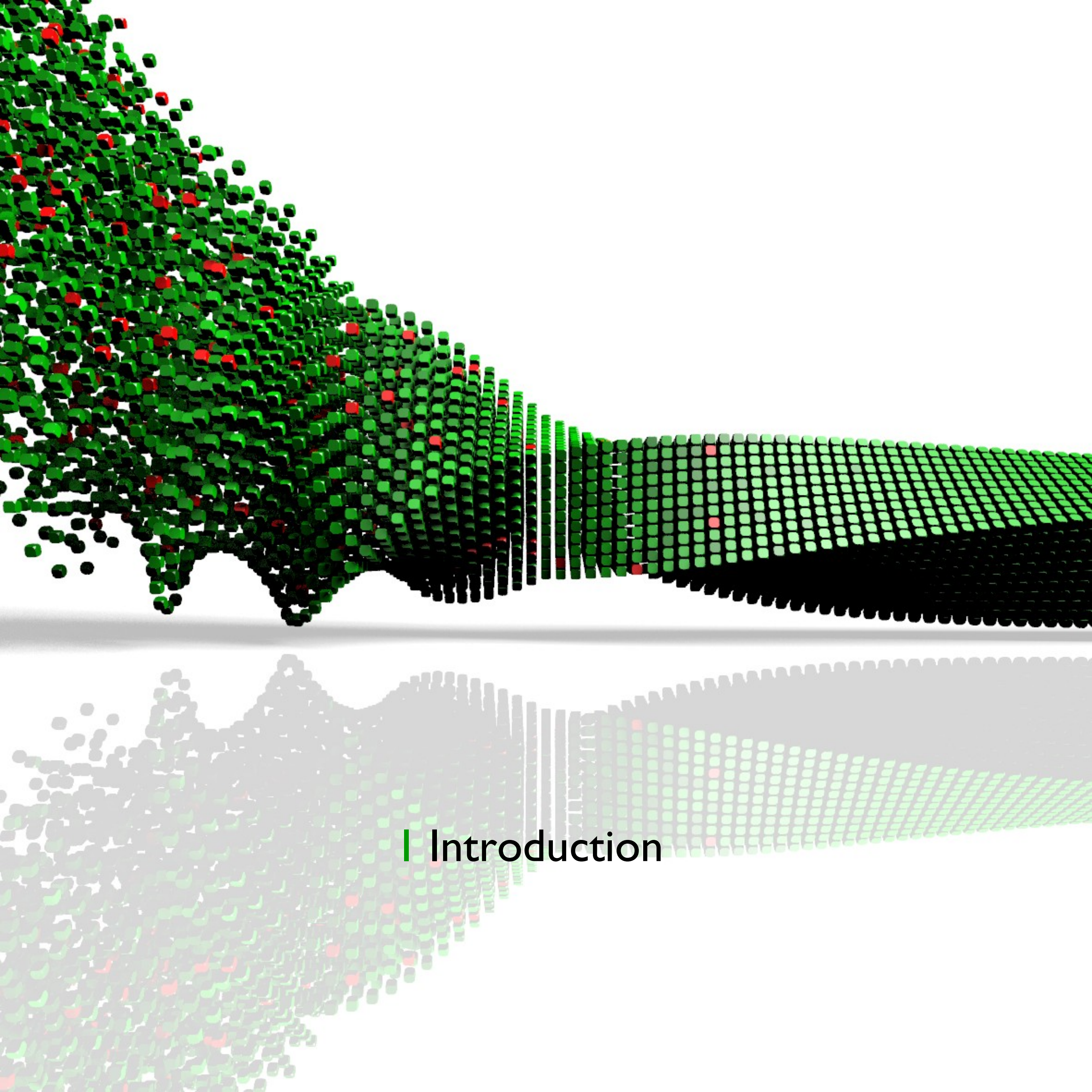
CONFORMIQ
Automated Test Design™

Designer 4.2

Evaluation Guide

Conformiq Evaluation Guide

Table of Contents

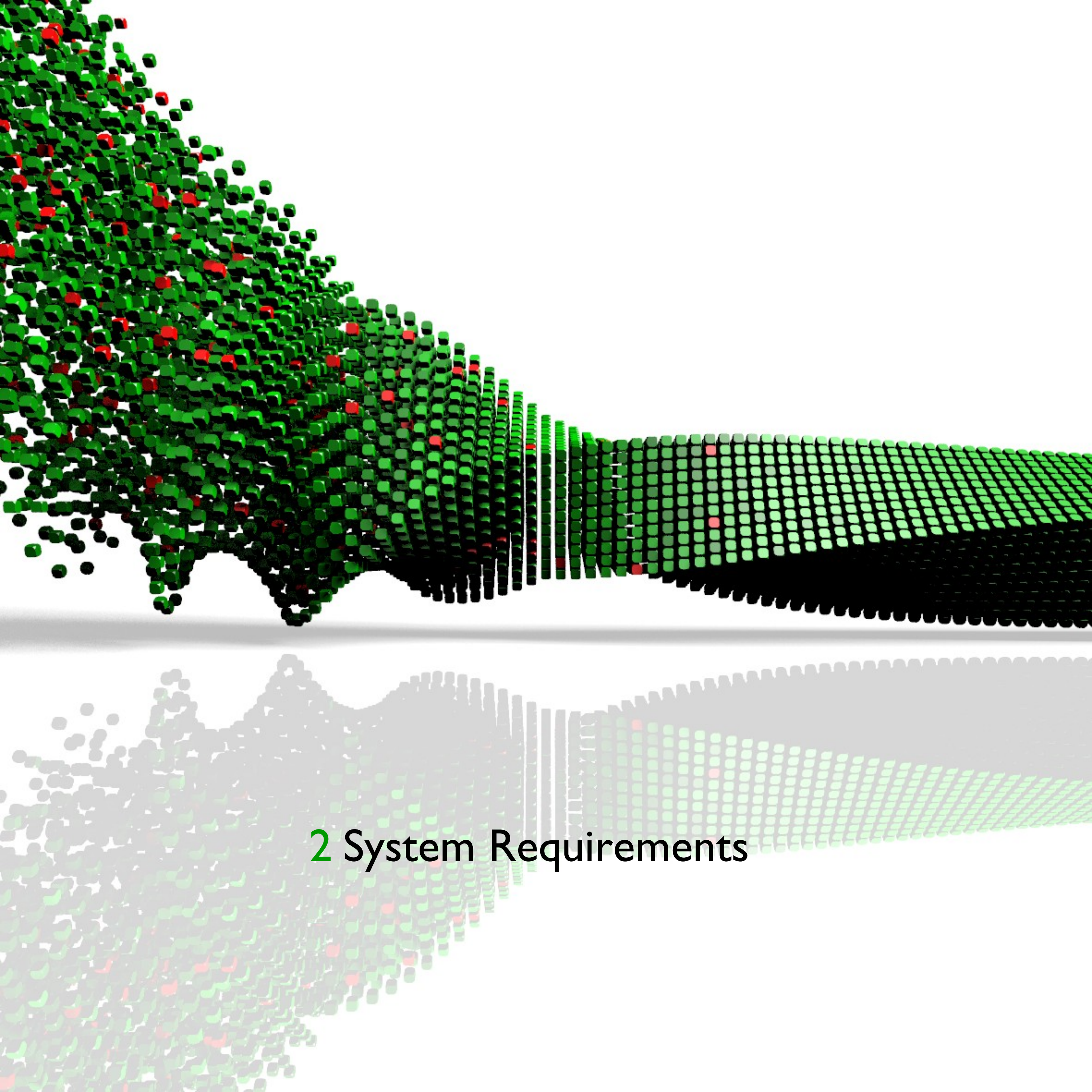


1 Introduction

This primer to Conformiq Designer and Conformiq Modeler exists to help you get started with your own copy.

The guide covers the following areas:

- How to install and setup Conformiq
- Conformiq Designer and Conformiq Modeler
- Test generation from a sample model
- Model changes and using Conformiq Designer with the modified model
- How to start your own Conformiq project



2 System Requirements

Conformiq Designer employs client-server architecture where the client user interface is implemented as an Eclipse plugin. The server component — Conformiq Computation Server — can be installed on the same computer as the Conformiq Eclipse Client or on another node on the local area network.

2.1 Conformiq Eclipse Client Requirements

Conformiq Eclipse Client is provided as

1. a standalone software as a rich client application that contains a minimal set of plug-ins collectively known as Rich Client Platform (RCP)
2. an Eclipse plugin that requires an existing Eclipse installation.

Conformiq RCP application and Conformiq Eclipse Client plugin versions are provided in two distinct installers.

If Conformiq Eclipse Client is installed as an Eclipse plugin, the required Eclipse must be *Eclipse 3.4 (Ganymede)* or newer. The recommended package is *Eclipse Classic*.

Shared requirements for both of the Conformiq Eclipse Client installation types are enumerated below:

- The required Java environment for running Conformiq Eclipse Client (QEC) is Sun Java 6 or higher.
- The system on which Conformiq Eclipse Client is installed should have at least 2048 MB memory.
- A relatively powerful CPU, a multiprocessor or multi-core processor computer is recommended.

2.2 Conformiq Computation Server Requirements

- Windows XP/Vista and most Linux distributions are supported by the Conformiq

Computation Server (QCS). It is highly recommended to install SP3 or newer to Windows XP in order to take advantage of the parallel test generation algorithm.

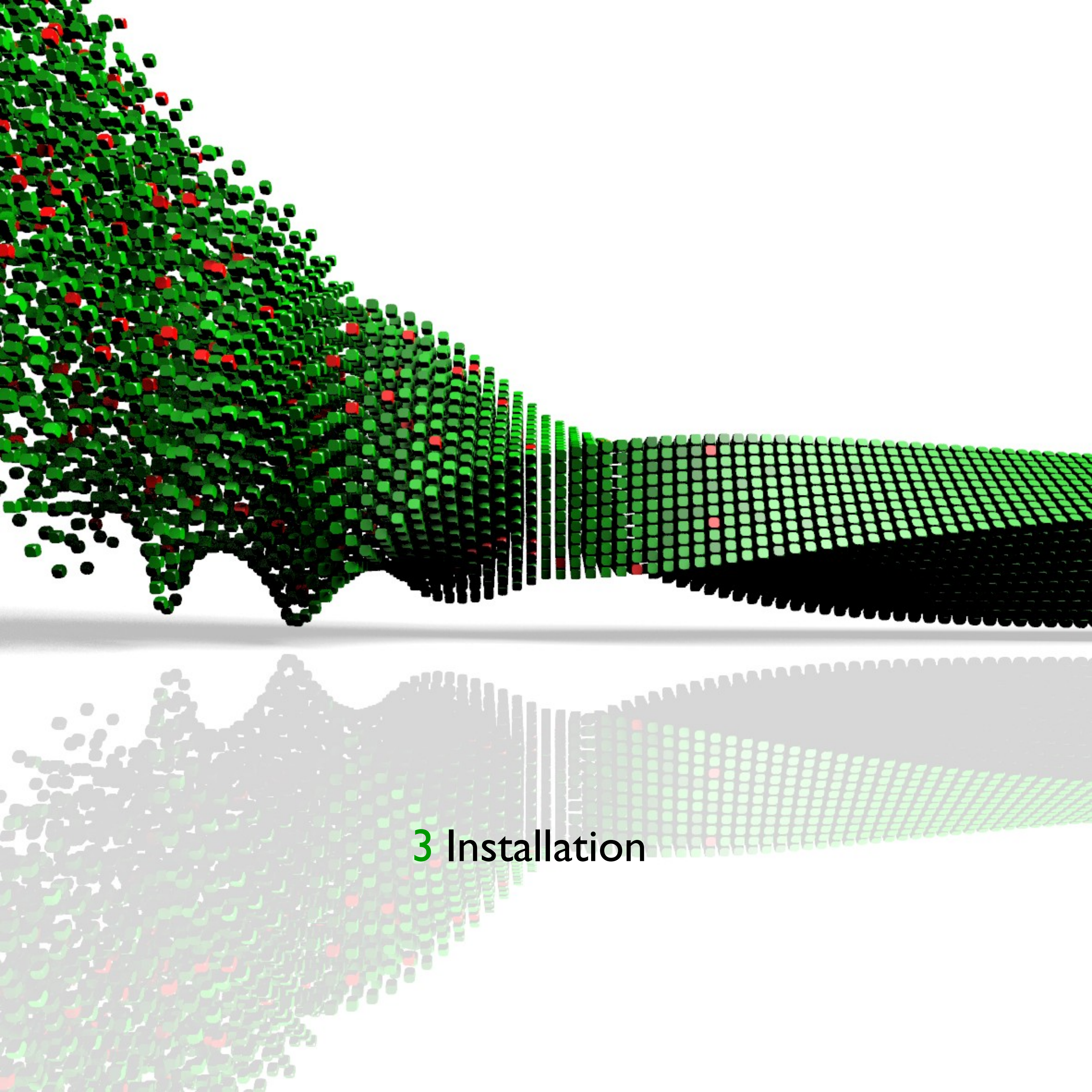
- The system on which Conformiq Computation Server is installed must have at least 2048 MB of memory but 4096 MB is recommended.
- We highly recommend a powerful computer with multiprocessor or multi-core processor that is Intel 586 (Pentium) compatible due to the large amount of calculations the software must do during automatic test generation.
- A 64 bit version is available for Linux.

2.3 Other Requirements

In addition, these software requirements are needed for a Linux installation:

- The GNU C Library (**libc** that defines "system calls" and other basic functionality) must be 2.4 or newer.

Test generation is a computationally intensive task and therefore it is recommended to run Conformiq Eclipse Client and Conformiq Computation Server on distinct computers. However, if QEC and QCS are both run on the same computer, the bare minimum amount of physical memory is 2048 MB but it is strongly recommended to have 4096 MB of memory or more and a powerful multiprocessor or multi-core processor.



3 Installation

Before installing Conformiq Eclipse Client, you should have Eclipse Ganymede or newer installed and functioning on your machine. To get Eclipse and for information on how to install it, please see Eclipse home page [http://www.eclipse.org].

Conformiq evaluation licenses enable that Conformiq can be evaluated for a limited time period. You should have received an evaluation license block from your distributor that needs to be copied into Eclipse (see instructions on how to do this below).

3.1 Windows

To install Conformiq, double-click on the "Conformiq <VERSION>.exe" file in Windows Explorer and follow the instructions step by step.

You can choose your directory of choice to install Conformiq but default is **C:\Program Files\Conformiq\Designer\**. It is suggested that the database is not installed in this same directory but to a location with normal user rights write permissions.

3.2 Linux

1. Choose the location where you want to install the software. This can be anywhere. If you want to make a system-wide installation a recommended location is /usr/local.
2. Make sure you have write permissions to the directory where you want to install the software. If you are doing a system-wide installation you may need to switch on to super-user privileges. If you do not have super-user privileges, you can always make a personal installation in your home directory.
3. Unpack the main distribution file. It is a file whose name starts with 'conformiq-' and ends with '.tgz', for example 'conformiq-4.2.0-linux-libc-2.4.tgz'. You have either downloaded this from a web page or you have it on your distribution CD. Give the command 'tar xzf (file)' where (file) is the distribution file.
4. After unpacking enter into the newly created directory whose name is, e.g.

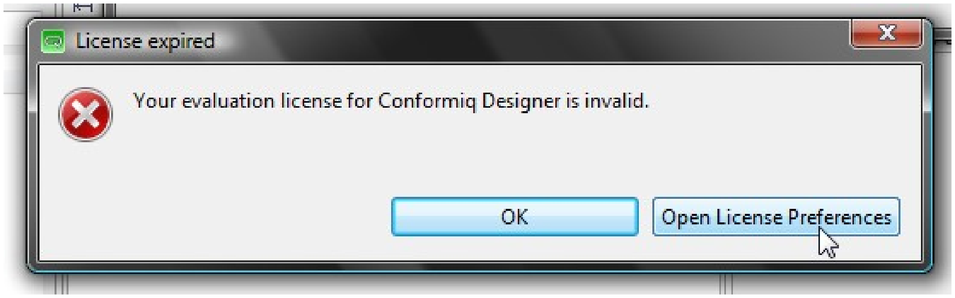
'conformiq-4.2.0-linux-libc-2.4'. Run the 'install.sh' shell script with suitable user privileges, e.g. '/bin/sh install.sh'. The installation script will ask you for installation information for Conformiq and test case database and the location of your Eclipse installation.

5. If install.sh notifies you about errors and or problems, try to remedy them and run install.sh again. If you cannot get past the errors, contact your supplier.
6. After running install.sh correctly, start first your computation server with ./conformiq-manager from your installation directory. After this you are ready to launch Eclipse to see your Conformiq perspective.

3.3 Setting Up Your Conformiq License

In order to use Conformiq you need a valid license for it. Please make sure you have a valid license before you try to use Conformiq (Note: Setting up a license server for your company is beyond the scope of this document. See User manual for those details.)

1. When you try to create your first Conformiq project, the following dialog appears:

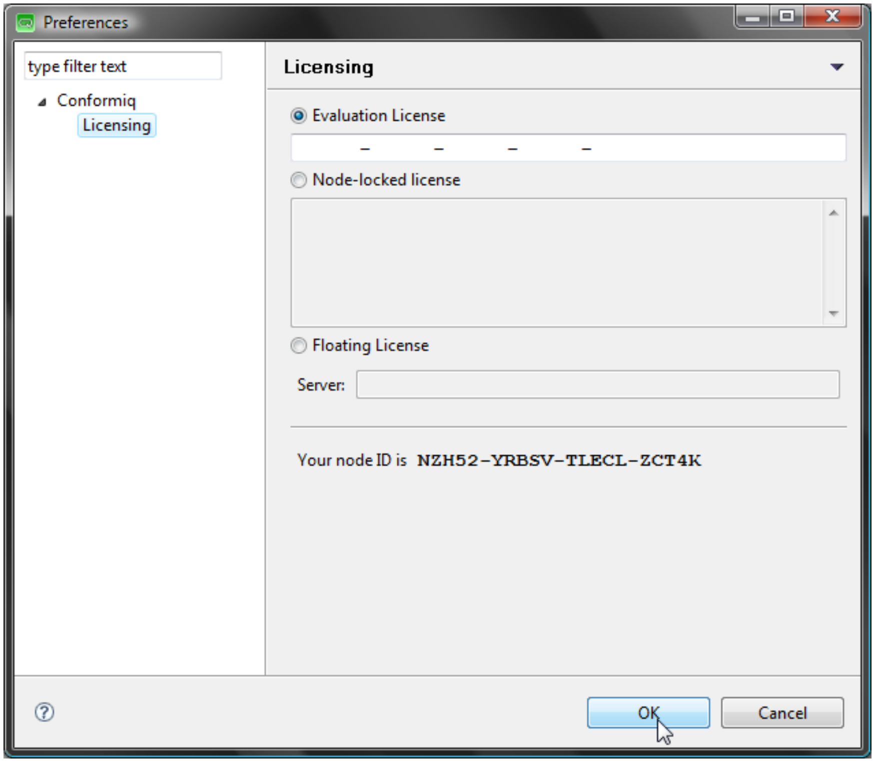


Your evaluation license for Conformiq Designer is invalid.

License has not been specified.

2. Just click on 'Open License Preferences' to set your evaluation license key into the 'Evaluation License' box. You should have received your evaluation license from

your distributor:



(●) Evaluation License

( ) Node-locked license

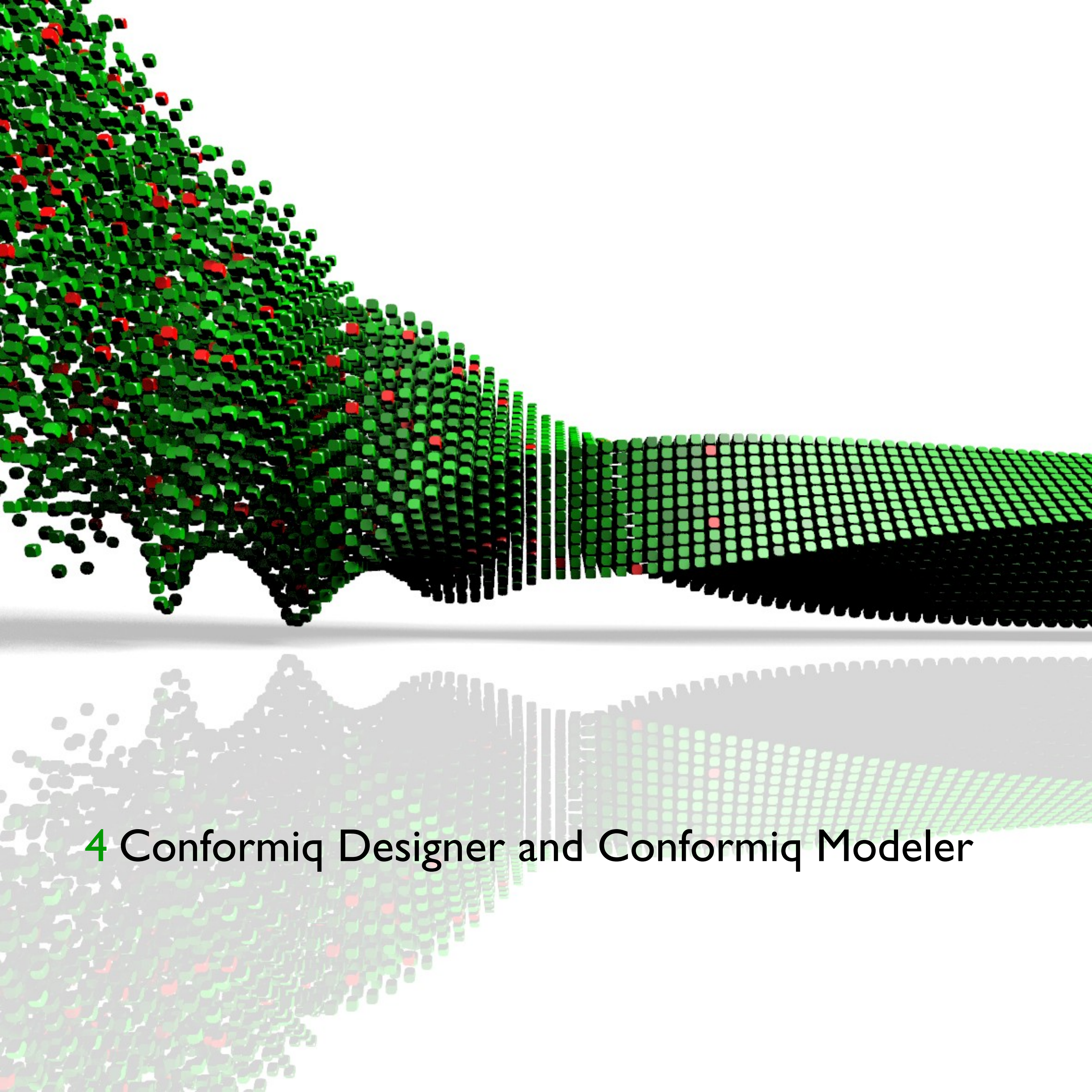
( ) Floating License

Server:

Your node ID is NZH52-YRBSV-TLECL-ZCT4K

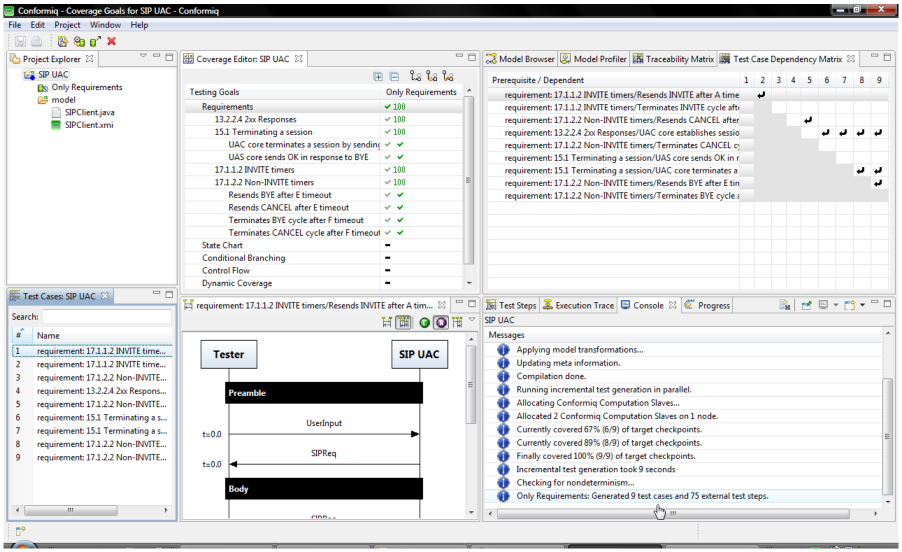
Add evaluation license key.

3. Enter you evaluation key
4. Click OK when you are done to continue creating your first model.



4 Conformiq Designer and Conformiq Modeler

Conformiq is a tool that designs test cases and test data automatically from a system model. System models, in accordance with specifications, can change over time. Using Conformiq makes it easy to generate the new test suite for the changed functionality. Test generation is controlled through a Conformiq perspective in Eclipse.



Testing Goals	Only Requirements
Requirements	✓ 100
13.2.2.4 2xx Responses	✓ 100
15.1 Terminating a session	✓ 100
UAC core terminates a session by sending	✓ ✓
UAS core sends OK in response to BYE	✓ ✓
17.1.1.2 INVITE timers	✓ 100
17.1.2.2 Non-INVITE timers	✓ 100
Resends BYE after E timeout	✓ ✓
Resends CANCEL after E timeout	✓ ✓
Terminates BYE cycle after F timeout	✓ ✓
Terminates CANCEL cycle after F timeout	✓ ✓
State Chart	-
Conditional Branching	-
Control Flow	-
Dynamic Coverage	-

Prerequisite / Dependent	1	2	3	4	5	6	7	8	9
requirement: 17.1.1.2 INVITE timers/Resends INVITE after A time		↵							
requirement: 17.1.1.2 INVITE timers/Terminates INVITE cycle aft									
requirement: 17.1.2.2 Non-INVITE timers/Resends CANCEL after					↵				
requirement: 13.2.2.4 2xx Responses/UAC core establishes sessio						↵	↵	↵	↵
requirement: 17.1.2.2 Non-INVITE timers/Terminates CANCEL c									
requirement: 15.1 Terminating a session/UAS core sends OK in r									
requirement: 15.1 Terminating a session/UAC core terminates a								↵	↵
requirement: 17.1.2.2 Non-INVITE timers/Resends BYE after E tin									↵
requirement: 17.1.2.2 Non-INVITE timers/Terminates BYE cycle a									

#	Name
1	requirement: 17.1.1.2 INVITE time...
2	requirement: 17.1.1.2 INVITE time...
3	requirement: 17.1.2.2 Non-INVITE...
4	requirement: 13.2.2.4 2xx Respons...
5	requirement: 17.1.2.2 Non-INVITE...
6	requirement: 15.1 Terminating a s...
7	requirement: 15.1 Terminating a s...
8	requirement: 17.1.2.2 Non-INVITE...
9	requirement: 17.1.2.2 Non-INVITE...

```
Messages
Applying model transformations...
Updating meta information.
Compilation done.
Running incremental test generation in parallel.
Allocating Conformiq Computation Slaves...
Allocated 2 Conformiq Computation Slaves on 1 node.
Currently covered 67% (6/9) of target checkpoints.
Currently covered 89% (8/9) of target checkpoints.
Finally covered 100% (9/9) of target checkpoints.
Incremental test generation took 9 seconds
Checking for nondeterminism...
Only Requirements: Generated 9 test cases and 75 external test steps.
```

Conformiq Designer generating tests from a system model.

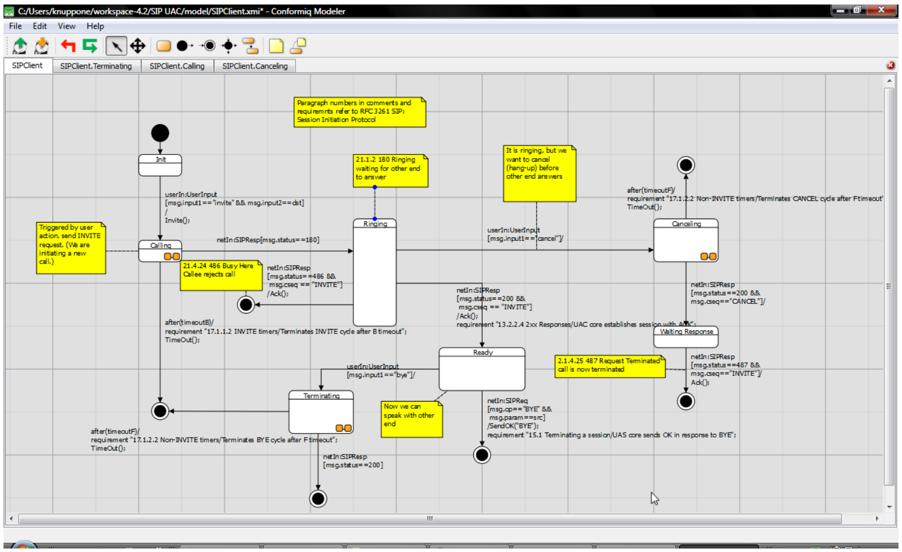
A system model can be created using 3rd party modeling tools or it can be created with Conformiq Modeling Language (QML) using the Conformiq Modeler.

QML allows the modeler to use both a textual and a graphical notation while describing the system:

- The textual notation of QML is a superset of the Java programming language. The textual part of the model can be created with any text editor.
- UML state charts are used as the graphical part of the model to describe the system

behavior. The graphical part is optional and the whole system could be described using QML textual representation only. However, use of the graphical notation is very useful and highly recommended for statefull systems.

The lightweight Conformiq Modeler comes bundled with Conformiq distribution allowing you to create fully functional models of your systems for automatic test case generation.



Paragraph numbers in comments and requiremnts refer to RFC 3261 SIP: Session Initiation Protocol

Triggered by user action, send INVITE request. (We are initiating a new call.)

21.1.2 180 Ringing waiting for other end to answer

21.4.24 486 Busy Here Callee rejects call

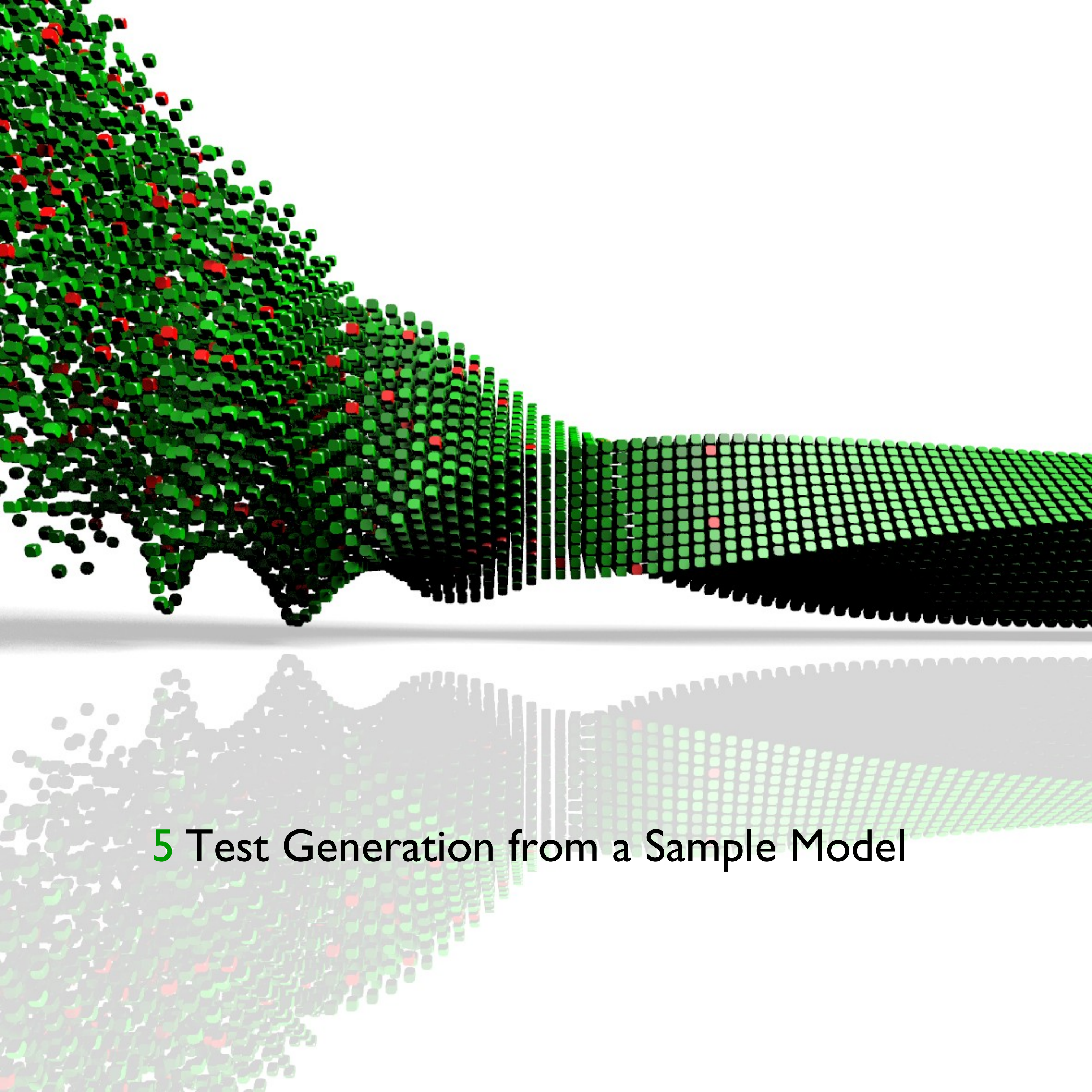
It is ringing, but we want to cancel (hang-up) before other end answers

2.1.4.25 487 Request Terminated call is now terminated

Now we can speak with other end

```
userIn:UserInput
[msg.input1=="invite" && msg.input2==dst]
/
Invite();

netIn:SIPResp[msg.status==180]

netIn:SIPResp
[msg.status==486 &&
 msg.cseq == "INVITE"]
/Ack();

after(timeoutB)/
requirement "17.1.1.2 INVITE timers/Terminates INVITE cycle after B timeout";
TimeOut();

userIn:UserInput
[msg.input1=="cancel"]/

after(timeoutF)/
requirement "17.1.2.2 Non-INVITE timers/Terminates CANCEL cycle after F timeout"
TimeOut();

netIn:SIPResp
[msg.status==200 &&
 msg.cseq == "INVITE"]
/Ack();
requirement "13.2.2.4 2xx Responses/UAC core establishes session with ACK";

netIn:SIPResp
[msg.status==200 &&
msg.cseq=="CANCEL"]/

netIn:SIPResp
[msg.status==487 &&
msg.cseq=="INVITE"]/
Ack();

userIn:UserInput
[msg.input1=="bye"]/

netIn:SIPReq
[msg.op=="BYE" &&
 msg.param==src]
/SendOK("BYE");
requirement "15.1 Terminating a session/UAS core sends OK in response to BYE";

after(timeoutF)/
requirement "17.1.2.2 Non-INVITE timers/Terminates BYE cycle after F timeout";
TimeOut();

netIn:SIPResp
[msg.status==200]
```

Conformiq Modeler



5 Test Generation from a Sample Model

This section highlights the necessary steps the user needs to take to generate tests from a system model.

5.1 Setting Up The Project

The model that will be used in this guide is a model of a SIP client. This example model can be found under the installation directory of Conformiq in the **Example Models/SIPClient** directory. In the following two different approaches are presented on how to take this SIP example model into use.

In preparation for using Conformiq Designer to generate tests from the model you will have to create a new Conformiq project in Eclipse that contains the model files.

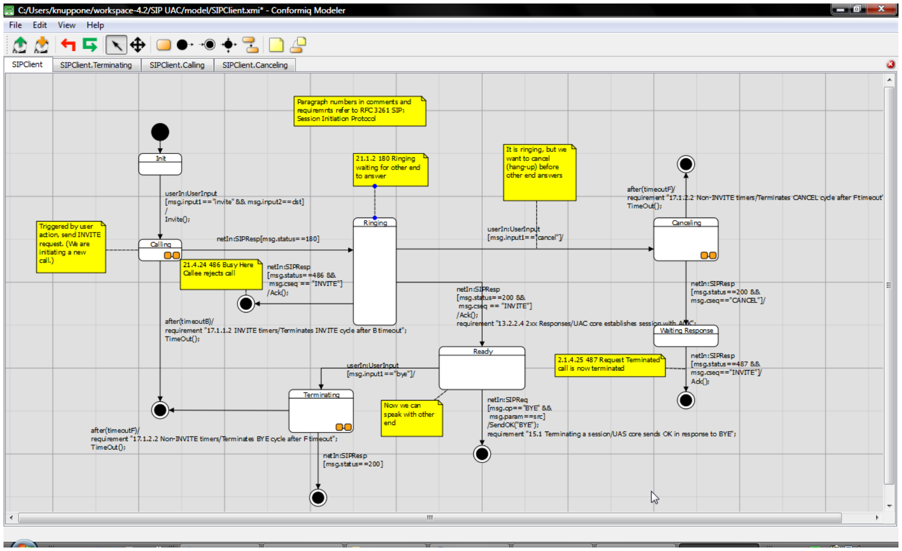
5.1.1 Importing Example Project

Conformiq Designer has a number of example projects that present different modeling patterns and approaches. Importing some of these example projects into Conformiq Designer Eclipse Client (QEC) is easy. To do so in QEC choose **File > New > Example...**. A number of example projects are presented where each presents some interesting modeling solution. For this presentation you should pick the **SIP UAC** from the list of available example projects and click on **Next** and **Finish**. If you are asked about whether you want to associate a specific perspective with this type of project, answer **Yes**. This will open all of the Conformiq Designer specific windows into QEC. That's it, you now have a ready Conformiq Designer project with all settings ready for starting test generation.

The "SIP UAC" model is composed of two model files that together compose the whole model. Note hod the model is composed of a textual (SIPClient.cqa) and a graphical part (SIPClient.xmi). These files will be next discussed in more details.

5.2 Using Conformiq Modeler

By double clicking on the SIPClient.xmi file in the newly imported example model, the model's state machine is opened in Conformiq Modeler.


C:/Users/knuppone/workspace-4.2/SIP UAC/model/SIPClient.xmi* - Conformiq Modeler
File Edit View Help
SIPClient
SIPClient.Terminating
SIPClient.Calling
SIPClient.Canceling
Paragraph numbers in comments and requiremnts refer to RFC 3261 SIP: Session Initiation Protocol
Init
userIn:UserInput [msg.input1=="invite" && msg.input2==dst] / Invite();
Triggered by user action, send INVITE request. (We are initiating a new call.)
Calling
netIn:SIPResp[msg.status==180]
21.1.2 180 Ringing waiting for other end to answer
Ringing
21.4.24 486 Busy Here Callee rejects call
netIn:SIPResp [msg.status==486 && msg.cseq == "INVITE"] /Ack();
after(timeoutB)/ requirement "17.1.1.2 INVITE timers/Terminates INVITE cycle after B timeout"; TimeOut();
It is ringing, but we want to cancel (hang-up) before other end answers
userIn:UserInput [msg.input1=="cancel"]/
after(timeoutF)/ requirement "17.1.2.2 Non-INVITE timers/Terminates CANCEL cycle after F timeout" TimeOut();
Canceling
netIn:SIPResp [msg.status==200 && msg.cseq=="CANCEL"]/
netIn:SIPResp [msg.status==200 && msg.cseq == "INVITE"] /Ack(); requirement "13.2.2.4 2xx Responses/UAC core establishes session with ACK";
Waiting Response
Ready
2.1.4.25 487 Request Terminated call is now terminated
netIn:SIPResp [msg.status==487 && msg.cseq=="INVITE"]/ Ack();
userIn:UserInput [msg.input1=="bye"]/
Terminating
Now we can speak with other end
netIn:SIPReq [msg.op=="BYE" && msg.param==src] /SendOK("BYE"); requirement "15.1 Terminating a session/UAS core sends OK in response to BYE";
after(timeoutF)/ requirement "17.1.2.2 Non-INVITE timers/Terminates BYE cycle after F timeout"; TimeOut();
netIn:SIPResp [msg.status==200]

Open SIPClient model in Conformiq Modeler

Conformiq Modeler is an included modeling tool supporting drawing State Charts that are a subset of UML. The usage of the tool is straightforward:

1. Pick some drawing tool from the tool bar.
2. Draw your system states and transitions on the canvas.
3. Add transition strings using QML textual notation to specify the behavior of the system as it moves from one state to another via a transition.
4. Model Element Tree provides a hierarchical view of the model.
5. You can zoom in and out arbitrarily using your mouse's wheel or as instructed in the View-menu.

The textual part of the model SIPClient.cqa can be viewed with Eclipse's built in text or Java editor. Using some Java editor enables you to have syntax high-lighting. The textual part of the model, written in Conformiq Modeling Language (QML) typically contains the data manipulation of the system model. It also contains reusable system methods and system attributes. This allows creating models where the graphical state machine depicts the high level behavior of the system whereas the textual part handles the system's data manipulation.

5.3 SIP Client Model Explained

5.3.1 System Description

This evaluation model is a partial model of a Session Initiation Protocol User Agent Client (SIP UAC). For this guide the system can be thought of as a VoIP client that communicates using the Session Initiating Protocol (SIP). The modeled functionalities of the client are session handling functionalities, in particular call setup, call termination and canceling call during call setup. Additionally the model takes into account timers associated to these functionalities. A real VoIP client has additional functionalities that has not been modeled here.

The model has been created based on the RFC 3261 specification, specifically chapter 17.1.2 that describes the session handling of the session handling protocol. This functionality has been modeled at a high level, leaving most of the SIP headers outside of the model. Handling of the missing fields in test execution is handled by the test harness.

The system has two test interfaces that are used by the model. The user interface is used to start the call creation and to inform the user of possible timeouts during session creation. The network interface is used to transmit the Session Initiation Protocol packets or messages.

From the system specification (RFC), the modeling engineer has retrieved requirements that have been included into the model.

1. Acknowledge established call (by sending ACK)
2. Acknowledge terminated call (by sending “200 OK” as a response to “BYE”)

3. Request retransmit timers for INVITE/non-INVITE requests (timer A, timer E)
4. Transaction timers for INVITE/non-INVITE transactions (timer B, timer F)

You can try to find these requirements in the model by searching for the 'requirement' keyword. For example the requirement for timer B timeout is modeled in the SIPClient.xmi state machine on the transition going from the 'Calling' state to the end state.

5.4 Generating Tests Using Conformiq Designer

5.4.1 Settings for Conformiq Designer Test Generation

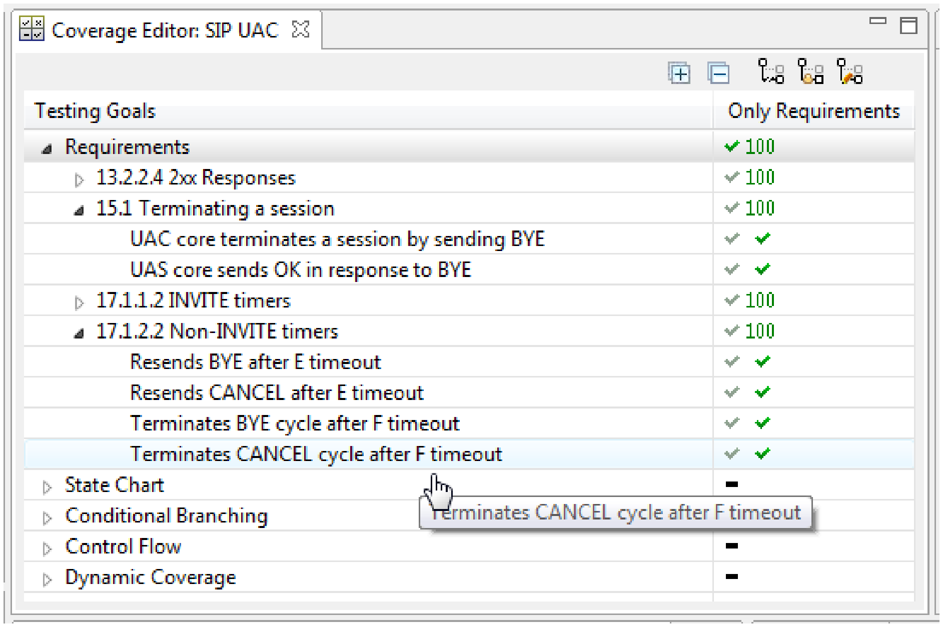
Before starting test case generation you should take a moment to check some settings.

By double clicking on the design configuration of your project (called "Only requirements") you will open a Coverage Editor. This coverage editor allows you to define which parts of the model you wish Conformiq Designer to traverse during the test generation. There is also the possibility to explicitly block parts of the model out of test generation. You would typically use the design configuration as a way to generate different types of test suites out of a same model.

The coverage editor thus allows you to change the nature of the test case generation. Each line represents a testing goal that Conformiq Designer will try to cover during test generation. Also as said earlier the testing goals can be set to different values.

- 'TARGET' meaning that Conformiq Designer will try to generate at least one test case that will cover this goal. By default 'Requirements', 'State Chart' and 'Control Flow/Methods' are set as target testing goals.
- The 'BLOCK' coverage goal means that Conformiq Designer will avoid generating tests that cover such a goal.
- 'DONT_CARE' goal means that no effort is put to cover this coverage goal. After test generation it can be covered but if it is not, it will not affect the total coverage information.

- Coverage goals are structured hierarchically. Setting a coverage goal to 'INHERIT', means that this goal should be a 'TARGET', 'BLOCK' or 'DONT_CARE' depending on the value set to this goal's parent.



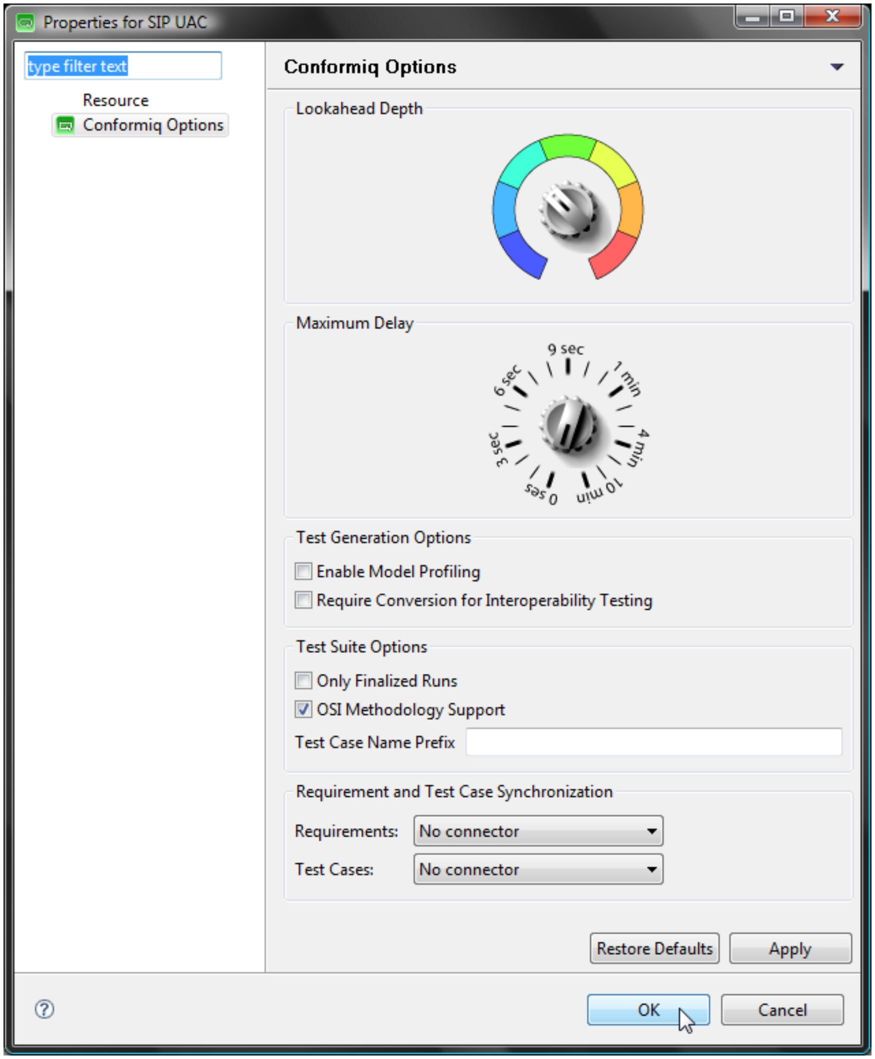
Testing Goals	Only Requirements
Requirements	✓ 100
13.2.2.4 2xx Responses	✓ 100
15.1 Terminating a session	✓ 100
UAC core terminates a session by sending BYE	✓ ✓
UAS core sends OK in response to BYE	✓ ✓
17.1.1.2 INVITE timers	✓ 100
17.1.2.2 Non-INVITE timers	✓ 100
Resends BYE after E timeout	✓ ✓
Resends CANCEL after E timeout	✓ ✓
Terminates BYE cycle after F timeout	✓ ✓
Terminates CANCEL cycle after F timeout	✓ ✓
State Chart	-
Conditional Branching	
Control Flow	-
Dynamic Coverage	-

Coverage Editor settings

You could have several design configurations in your project, allowing you to generate different types of test suites.

While the testing goals defined in the Coverage Editor are specific to a test design configuration it is possible to influence on the depth of test generation at the project level. This is done modifying the 'Lookahead Depth' that is located in the properties of the 'SIP UAC' project. Right click on the project and choose **Properties > Conformiq Options**. Decreasing the value of this knob will make covering all of the coverage options impossible. Increasing this knob will not find new test cases, but test generation time will increase because

Conformiq Designer needs to do a deeper test search.


Properties for SIP UAC

type filter text

Resource

Conformiq Options

Conformiq Options

Lookahead Depth

Maximum Delay

Test Generation Options

[ ] Enable Model Profiling

[ ] Require Conversion for Interoperability Testing

Test Suite Options

[ ] Only Finalized Runs

[✓] OSI Methodology Support

Test Case Name Prefix

Requirement and Test Case Synchronization

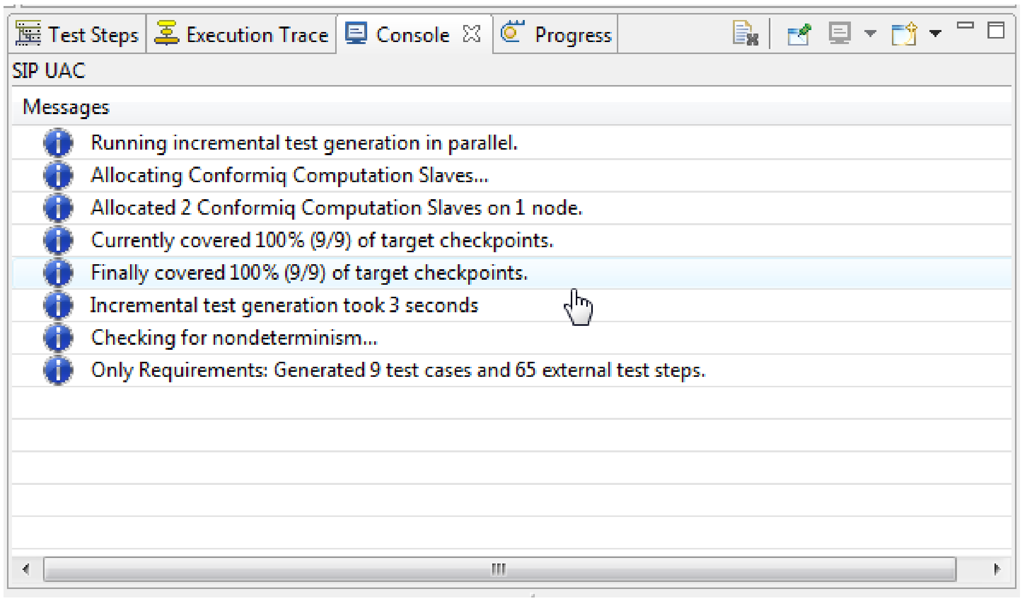
Requirements: No connector

Test Cases: No connector

Correct Lookahead value

5.4.2 Generating Tests

Using the project's default coverage settings you can start the test case generation. Test generation ends when the console window reports "INFO: 100% coverage reached, stopping.". Conformiq Designer has now generated test cases for all the set testing goals defined in the coverage editor.



SIP UAC

Messages

```
Running incremental test generation in parallel.
Allocating Conformiq Computation Slaves...
Allocated 2 Conformiq Computation Slaves on 1 node.
Currently covered 100% (9/9) of target checkpoints.
Finally covered 100% (9/9) of target checkpoints.
Incremental test generation took 3 seconds
Checking for nondeterminism...
Only Requirements: Generated 9 test cases and 65 external test steps.
```

All testing goals covered

5.4.3 Viewing the Output

Once the tests have been created it is good practice to analyze the tests. In the following different ways of analyzing the generated test cases are presented.

Specific test cases can be selected for inspection either directly from the 'Test Case List' window or from the 'Traceability Matrix' window. The active view can be changed from the menu under **Window > Show View**.

The **Test Case List** has simply the list of all generated test cases numbered from first to last. You can here select a test case and view its behavior in the 'Test Case' window. You can also rename test cases to give them more significant names.

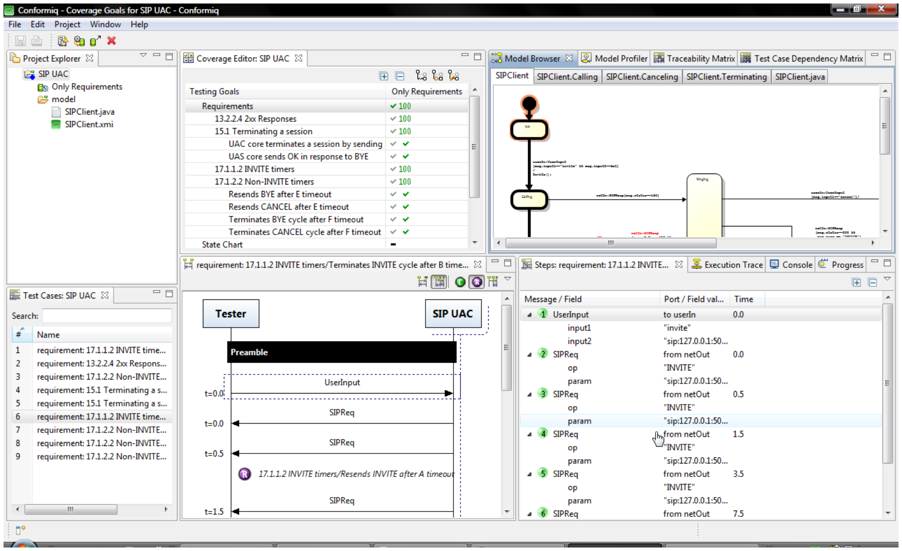
The **Traceability Matrix** allows you to see the different testing goals linked with the test case number that covers this goal. This allows you to trace which test case covers which requirement (or other testing goal).

The **Test Case** window presents a test case as a Message Sequence Chart (MSC) where input and output messages are drawn between the 'tester' and the 'SIPClient' lifelines. An arrow going from the 'Tester' to the 'SIPClient' means that the test environment is sending an input to the system and respectively an arrow leaving the 'SIPClient' means that the system has sent out a response to be validated against the expected value.

The **Test Steps** window shows the payload data used in the active test case at a chosen test step. These values are automatically generated from the system model.

Find the test case that covers the requirement "Terminates INVITE cycle after B timeout" using the 'Traceability Matrix'. The MSC of this test case describes the steps to test this requirement.

1. Initially the 'Tester' sends an input message through the SIPClient's userIn interface.
2. The test harness should then expect the SIPClient (named 'SIP UAC' in the MSC) to send a 'SIPReq' request message on its netOut interface.
3. The 'SIPReq' message should be perceived four times in total from the netOut interface.
4. Lastly a timeout indication message should be sent to the user with a 'TimeOutIndication' message perceived on the userOut interface.



Testing Goals	Only Requirements
Requirements	✓ 100
13.2.2.4 2xx Responses	✓ 100
15.1 Terminating a session	✓ 100
UAC core terminates a session by sending	✓ ✓
UAS core sends OK in response to BYE	✓ ✓
17.1.1.2 INVITE timers	✓ 100
17.1.2.2 Non-INVITE timers	✓ 100
Resends BYE after E timeout	✓ ✓
Resends CANCEL after E timeout	✓ ✓
Terminates BYE cycle after F timeout	✓ ✓
Terminates CANCEL cycle after F timeout	✓ ✓
State Chart	-

#	Name
1	requirement: 17.1.1.2 INVITE time...
2	requirement: 13.2.2.4 2xx Respons...
3	requirement: 17.1.2.2 Non-INVITE...
4	requirement: 15.1 Terminating a s...
5	requirement: 15.1 Terminating a s...
6	requirement: 17.1.1.2 INVITE time...
7	requirement: 17.1.2.2 Non-INVITE...
8	requirement: 17.1.2.2 Non-INVITE...
9	requirement: 17.1.2.2 Non-INVITE...

Message / Field	Port / Field val...	Time
1 UserInput	to userIn	0.0
input1	"invite"	
input2	"sip:127.0.0.1:50...	
2 SIPReq	from netOut	0.0
op	"INVITE"	
param	"sip:127.0.0.1:50...	
3 SIPReq	from netOut	0.5
op	"INVITE"	
param	"sip:127.0.0.1:50...	
4 SIPReq	from netOut	1.5
op	"INVITE"	
param	"sip:127.0.0.1:50...	
5 SIPReq	from netOut	3.5
op	"INVITE"	
param	"sip:127.0.0.1:50...	
6 SIPReq	from netOut	7.5

Test case message sequence chart

5.4.4 Exporting Test Cases

Conformiq allows you to export the designed test cases out of Conformiq Designer, using publishers or scripters as they are also sometimes called. A publisher is an application that translates the test cases from the Conformiq Designer internal representation to a predefined syntax. This syntax is the one that is used by your test execution adaptation.

Under the tool's installation directory there are several default publishers that are provided. Using these it is possible to export the designed test cases into HTML, TTCN-3, Perl or TCL output. It is even possible to write the tests out to HP QualityCenter9.

Adding a publisher to a project is done by right clicking the project's design configuration and selecting **New > Scripting Backend**. In the installation directory you can find the 'Example Scripters' directory that contains the above mentioned scripters.

5.4.5 Changing Test Coverage Settings

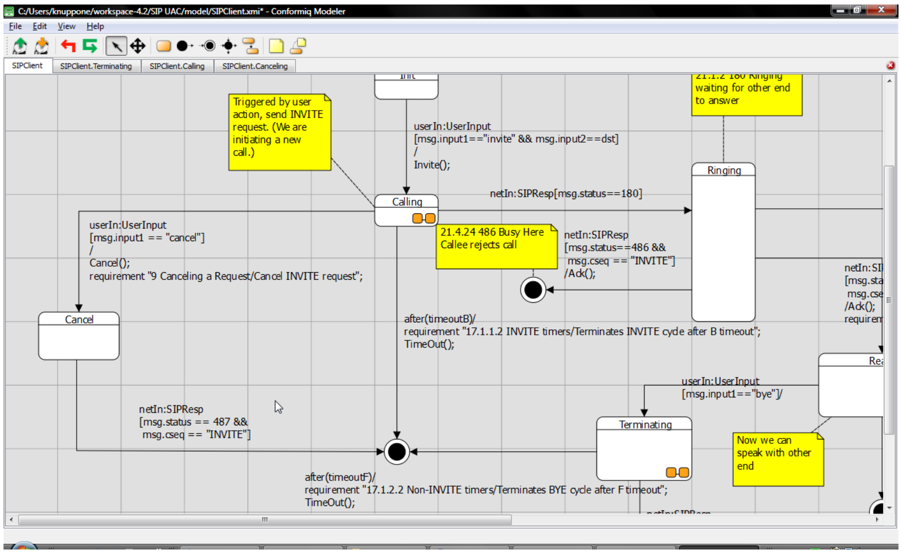
As mentioned above you can modify the test generation settings from the design configuration's 'Coverage Editor'. Lets try to generate a more thorough test suite by requiring that Conformiq Designer also covers all structural features of the model. This means we want a test suite that in addition to the requirements covers also all states and transitions of the model too. This functionality is controlled by the 'State Chart - States' and 'State Chart - Transitions' coverage goals.

To change these settings open the 'Coverage Editor' by double clicking on the design configuration of the project. It is called 'Only Requirements'. Change under 'State Chart' the goals for 'State' and 'Transitions' from 'DON'T CARE' to 'TARGET' and restart test generation. It is this simple to change test generation coverage options. Your test suite will now consist of 16 test cases instead of the original 9.

5.5 Modifying the Model

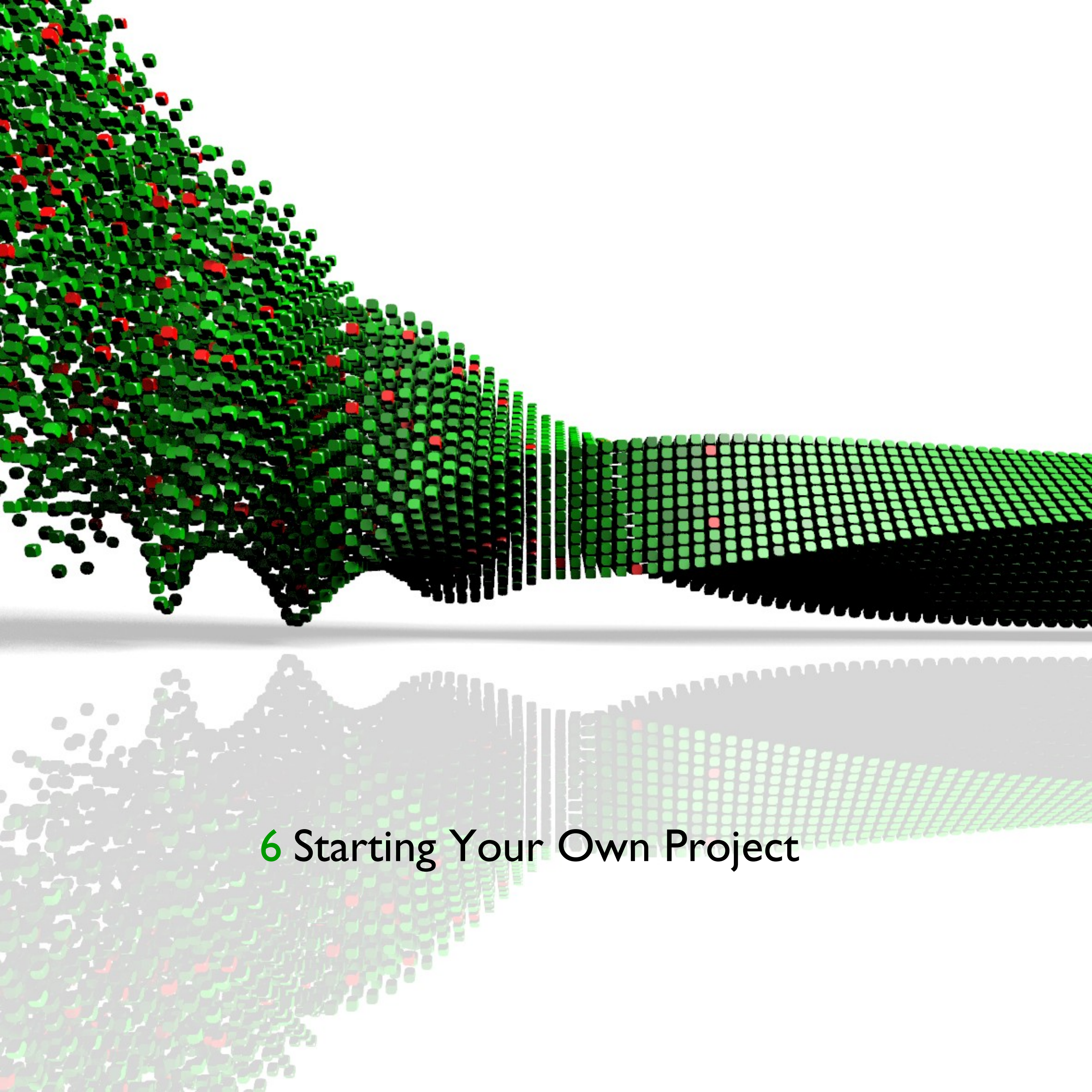
The presented model in this guide does not model the full functionality of a SIP client. We can then continue modeling and add it new functionality as defined in the system's specification. The RFC based on which the model has been created tells us how an invite request can be canceled, in chapter '9 Canceling a Request'.

```
[...] For this reason, CANCEL is best for INVITE requests, which
can take a long time to generate a response. In that usage, a UAS
that receives a CANCEL request for an INVITE, but has not yet sent a
final response, would "stop ringing", and then respond to the INVITE
with a specific error response (a 487).
```


C:/Users/knuppone/workspace-4.2/SIP UAC/model/SIPClient.xmi* - Conformiq Modeler
File Edit View Help
SIPClient
SIPClient.Terminating
SIPClient.Calling
SIPClient.Canceling
Triggered by user action, send INVITE request. (We are initiating a new call.)
userIn:UserInput [msg.input1=="invite" && msg.input2==dst] / Invite();
21.1.2 180 Ringing waiting for other end to answer
Ringing
netIn:SIPResp[msg.status==180]
Calling
userIn:UserInput [msg.input1 == "cancel"] / Cancel(); requirement "9 Canceling a Request/Cancel INVITE request";
21.4.24 486 Busy Here Callee rejects call
netIn:SIPResp [msg.status==486 && msg.cseq == "INVITE"] /Ack();
netIn:SI [msg.sta msg.cse /Ack(); requiren
Cancel
after(timeoutB)/ requirement "17.1.1.2 INVITE timers/Terminates INVITE cycle after B timeout"; TimeOut();
Rea
userIn:UserInput [msg.input1=="bye"]/
netIn:SIPResp [msg.status == 487 && msg.cseq == "INVITE"]
Terminating
Now we can speak with other end
after(timeoutF)/ requirement "17.1.2.2 Non-INVITE timers/Terminates BYE cycle after F timeout"; TimeOut();

Added canceling of invite request functionality in Conformiq Modeler

Once you have added this functionality to your model and saved it, you are ready to start test generation. You should now receive one additional test case that tests this added functionality.

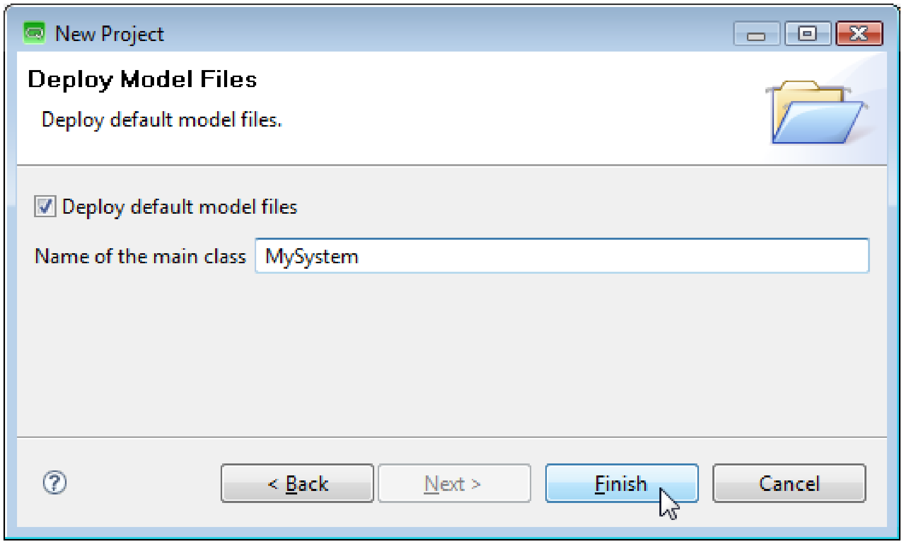


6 Starting Your Own Project

You've now learned the basics of Conformiq in terms of a sample project. The best way to get an in-depth understanding is to use Conformiq in a project of your own. Below you'll find some words of advice to assist you on this exciting journey.

1. The key thing in using Conformiq is aligning your thinking with that of a system designer: rather than considering how to test the system you put yourself into the shoes of the designer and concentrate on how the system works. What you then end up with in the form of a model is an abstract implementation of the system to be used as a basis for automatic test design.
2. As an implication of the above you should note that everything you've learned in your career about system architecture, data encapsulation, separation of concerns, OO design patterns etc. is of high value. Use this knowledge to come up with high quality models.
3. Consider the test interface of your system always first: this is important because the level of detail in the test interface essentially defines the abstraction level of your model.
4. Start with a simple "pipe cleaner" model to ensure that the pipeline from the model through test generation to test execution works.
5. Work in increments; rather than trying to put all the functionality of the system into the very first version of the model build the model one functionality after another.
6. Have your peers review your models.
7. Use 'requirements' in the model to allow you to easily control and track the coverage of the generated test cases with respect to different functionalities expressed in the model.

To create your own project use the 'Conformiq Project' wizard. In Conformiq Eclipse Client click **File > New > Conformiq Project** and fill in your project name before pressing **Next**. Make sure to check the 'Deploy default model files' option and fill in the name of your main test component.


New Project

Deploy Model Files

Deploy default model files.

☑ Deploy default model files

Name of the main class: MySystem

Create a new project and deploy skeleton model files